# セキュリティ機構のオフロード時の性能分離

新井 昇鎬　　光来 健一　　千葉 滋

本論文では、セキュリティ機構をオフロードした場合でも仮想マシン間の性能分離を実現するシステム OffloadCage を提案する。攻撃を受ける可能性のある仮想マシンから別の仮想マシンや仮想マシンモニタにセキュリティ機構をオフロードすると、セキュリティ機構自体が攻撃を受けにくくなりセキュリティは向上する。しかし、オフロードしたセキュリティ機構の資源の使用量はオフロード元の仮想マシンにカウントされないため、仮想マシン間の資源分配に問題が生じる。OffloadCage はオフロードしたセキュリティ機構の資源の使用量を計測して、オフロード元の仮想マシンが使ったものとしてスケジューリングを行う。我々は 2 種類のセキュリティ機構をオフロードして実験を行い、オフロードしたセキュリティ機構とオフロード元の仮想マシンの資源の使用量の合計がオフロード元の仮想マシンに設定された制限を守れていることを確認した。

## 1　はじめに

近年、サーバの管理コストの削減や利用率の向上の手段として仮想マシン環境が注目されている。仮想マシンを用いることで、複数のサーバマシンを一つのマシンに集約することができる。この際に、仮想マシン間の性能は分離できるようにするべきである。ある仮想マシンの資源の消費が、同じ物理マシン上の他の仮想マシンの性能に影響を与えるべきではない。もし影響を与えてしまうと、他の仮想マシンが十分な品質のサービスを提供できなくなる可能性があるためである。

しかし、セキュリティ機構のオフロードを行うと、このような性能の分離をうまく行うことができない。セキュリティ機構のオフロードとは、攻撃を受ける可能性がある仮想マシンからセキュリティ機構を外へ出し、別の仮想マシンや仮想マシンモニタで動作させることである。オフロードを行うことにより、セキュリティ機構自体が攻撃されにくくなりセキュリティが向上する。その一方で、オフロードしたセキュリティ機構の資源の消費により性能の分離をうまく行うことができなくなる。オフロードしたセキュリティ機構の資源の消費量はオフロード元の仮想マシンにカウントされないため、合計の資源の消費量はオフロード元の仮想マシンに設定された制限を越えてしまう。

本論文では、セキュリティ機構をオフロードした場合でも仮想マシン間の性能の分離を実現するシステム OffloadCage を提案する。OffloadCage はセキュリティ機構を監視する OC-Monitor とオフロードを意識したスケジューリングを行う OC-Scheduler から構成される。OC-Monitor はオフロードしたセキュリティ機構の資源の使用量を定期的に OC-Scheduler に通知する。OC-Scheduler は Xen [1] の既存のスケジューラであるクレジットスケジューラ [2] をベースとして、オフロードしたセキュリティ機構とオフロード元の仮想マシンの CPU 使用量の合計に基づいたスケジューリングを行う。これにより、セキュリティ機

Sungho Arai, Sigeru Chiba, 東京工業大学, Tokyo Institute of Technology
Kenichi Kourai, 九州工業大学, Kyushu Institute of Technology, 科学技術振興機構, CREST, Japan Science and Technology Agency

構をオフロードした場合でも使用できる資源の量を正しく制限でき、他の仮想マシンの資源に性能を与えないようにすることができる。

OffloadCage により性能がうまく分離出きるかどうかを確かめるために、2 種類のセキュリティ機構をオフロードした場合の性能について調べる実験を行った。セキュリティ機構としては、パケット受信のイベント毎にチェックを行う snort [3] と一定時間毎にファイルシステムのチェックを行う tripwire [4] を用いた。この実験により、オフロードしたセキュリティ機構とオフロード元の仮想マシンが使用した CPU 資源の合計が、オフロード元に設定された CPU 資源に関する制限をほぼ守れていることを確認した。

以下、2 章では仮想マシンを利用したセキュリティ機構のオフロードの利点と問題点について述べる。3 章では OffloadCage の設計と実装について述べ、4 章では OffloadCage による性能分離を確かめる実験について述べる。5 章では関連研究、6 章で本論文をまとめる。

## 2 セキュリティ機構のオフロード

仮想マシンを利用してセキュリティ機構のオフロードを行うことにより、セキュリティを向上させることができる (図 1 参照)。外部にサービスを提供しており攻撃を受ける可能性のある仮想マシンからセキュリティ機構を外へ出し、別の仮想マシンや仮想マシンモニタで動作させることで、セキュリティ機構自体を攻撃から守ることができる。仮想マシンモニタの機能を使うことで、オフロード元の仮想マシンに対してセキュリティのチェックを行うことができる。

たとえば、仮想マシンを利用して Snort や Tripwire をオフロードする場合を考える。Snort はネットワーク型の侵入検知システムであり、監視対象のホストが送受信するネットワークパケットを監視して、シグネチャと呼ばれるルールと比較することにより攻撃を検出する。オフロードされた Snort は監視対象の仮想マシンのネットワークインターフェイスを監視することで、仮想マシンのパケットを取得することができる。一方、Tripwire はホスト型の侵入検知システムである。定期的にファイルのメタ情報や内容などの

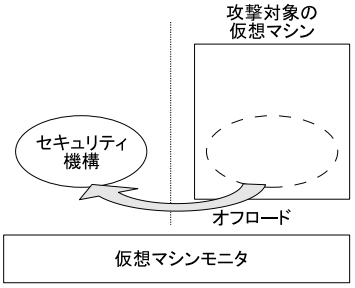


**図 1　仮想マシンを利用したオフロード**

ファイルシステムの状態を監視してファイルシステムの不整合を検出する。オフロードされた Tripwire は監視対象の仮想マシンが使う仮想ディスクの内容を参照することで、整合性がとれている時のファイルシステムの状態と比較することができる。

セキュリティ機構のオフロードは攻撃者が Snort や Tripwire のプロセス自体を直接攻撃することを防ぐだけでなく、Snort や Tripwire が使うポリシやデータの改竄も防ぐことができる。監視対象の仮想マシン内の OS 等に脆弱性があった場合、システムに侵入されてしまうかもしれない。このとき、Snort のシグネチャを書き換えたり Tripwire が使う整合性のとれたのファイルシステムの状態を保存してあるデータベースを改竄することで、攻撃を検知されるのを防ぐことができる。しかし、セキュリティ機構をオフロードしていれば、シグネチャやデータベースも仮想マシンの外のものを使用するため、ポリシの変更やデータベースの改竄を防ぐことができる。

しかし、セキュリティ機構をオフロードすると仮想マシン間の性能分離がうまく行えなくなる。各仮想マシンに分配された資源の量を守ることができている時に、仮想マシン間の性能は分離できていると言うことができる。オフロードしたセキュリティ機構の資源の消費量はオフロード元の仮想マシンにカウントすることができないため、オフロードしたセキュリティ機構とオフロード元の仮想マシンの資源の使用量の合計が、オフロード元の仮想マシンに割り当てられた分配を越えてしまう場合がある。分配を越えた分は他

の仮想マシンの性能に影響を与える恐れがある。

Snort をオフロードした場合、監視対象であるオフロード元の仮想マシンに大量のリクエストが送られたり、DoS 攻撃されたりした時に性能分離に大きな問題が生じる。例えば、オフロード元の仮想マシンに CPU 使用率 40 %の制限をかけているとする。この仮想マシンに大量のパケットが送られると、オフロードした Snort も処理が増え負荷がかかる。オフロードした Snort が CPU 資源を 30 %使用したとすると、オフロード元の仮想マシンのために合計で 70 %の CPU 資源が利用されたことになる。もし、CPU 資源を 50 %割り当てられた他の仮想マシンがあったとすると、オフロードの影響によりその仮想マシンは 50 %利用することができない。Tripwire をオフロードした場合は、Tripwire が検査を行っている間だけ、性能分離の問題が生じる。

## 3 OffloadCage

セキュリティ機構をオフロードした時の仮想マシン間の性能分離を行うシステム OffloadCage を提案する。OffloadCage では、オフロードしたセキュリティ機構の資源の使用量をオフロード元の仮想マシンが使用したものとしてカウントする。OffloadCage は Xen に実装し、ドメイン U と呼ばれる一般の仮想マシンからドメイン 0 と呼ばれる特権を持った仮想マシンにセキュリティ機構をオフロードする。以後、オフロード元であるドメイン U のことを仮想マシンと呼び、オフロード先であるドメイン 0 とは区別する。

OffloadCage は、図 2 のように OC-Monitor と OC-Scheduler から構成される。OC-Monitor はセキュリティ機構がオフロードされたドメイン 0 で動作し、セキュリティ機構が使用した資源の量を計測する。オフロード元の仮想マシンのスケジューリングに利用するために計測した値を OC-Scheduler に定期的に通知する。一方、OC-Scheduler は、オフロードしたセキュリティ機構の資源の使用量をオフロード元の仮想マシンで使われたものとしてスケジューリングを行う。

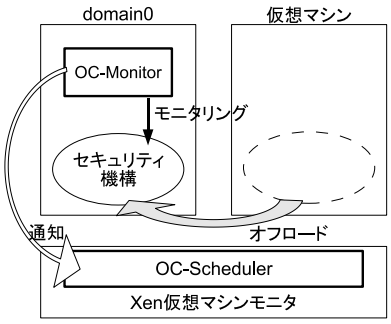


**図 2　アーキテクチャ**

### 3.1 OC-Monitor

OC-Monitor はオフロードしたセキュリティ機構が使用した CPU 資源を監視する。オフロードしたセキュリティ機構のプロセス ID を指定することで OC-Monitor は監視するプロセスを特定する。また、オフロードしたセキュリティ機構とオフロード元の仮想マシンを関連付けるために、同時に仮想マシンの ID も指定する。

OC-Monitor はセキュリティ機構が使用した CPU 資源量から CPU 使用率を計算する。CPU 使用率は一定時間にセキュリティ機構が使ったユーザ時間とカーネル時間の合計から計算することができる。累積の CPU 時間は/proc/プロセス ID/stat から取得できるため、前の区間で取得した CPU 時間との差から一定時間に使った CPU 時間を計算することができる。

OffloadCage はオフロードしたセキュリティ機構の CPU 利用率を OC-Monitor から OC-Scheduler に通知するためのハイパーコールを提供している。ハイパーコールは仮想マシンが仮想マシンモニタの機能を利用するために提供されている機構である。このハイパーコールは引数としてドメイン ID と CPU 利用率を取り、オフロードしたセキュリティ機構の CPU 利用率の情報を指定した仮想マシンに与える。

### 3.2 OC-Scheduler

OC-Scheduler はオフロードしたセキュリティ機構とオフロード元の仮想マシンの資源の使用量の合計に基づいてスケジューリングを行うスケジューラであ

る。OC-Scheduler により、あたかもセキュリティ機構がオフロードされていないかのように資源の分配を行うことができる。このスケジューラは Xen の仮想マシンスケジューラであるクレジットスケジューラを改良して作成した。

#### 3.2.1 クレジットスケジューラ

Xen の仮想マシンスケジューラは各仮想マシンに一つ以上の VCPU を割り当てる。VCPU とは仮想マシンが持つ仮想的な CPU である。VCPU を物理 CPU のランキューに入れてスケジューリングを行う。OS はプロセスをランキューに入れてスケジューリングを行うが、Xen では VCPU をランキューに入れてスケジューリングを行う。

クレジットスケジューラは Xen の現在のデフォルトの仮想マシンスケジューラである。このスケジューラでは各仮想マシンの VCPU に配布する CPU 時間をクレジットとして表現する。10ms ごとに現在実行状態にある VCPU のクレジットが減らされる。30ms ごとに各仮想マシンに設定された cap と weight を基にクレジットが計算、配布される。cap は仮想マシンが使用できる CPU 利用率を表す絶対的な値である。weight は仮想マシン間の CPU 資源の分配の比率を示す相対的な値である。たとえば、cap と weight が図 3 のように設定されているとする。このとき、30ms ごとに配布されるクレジットは weight により平等に配布される。しかし、cap が 40 の仮想マシンは最大で 40 %までしか CPU を利用できない。cap が 0 の仮想マシンは最大 CPU 利用率に関する制限がない。

また、各 VCPU には UNDER、OVER と BOOST という優先度がつけられる。BOOST が一番優先度が高く、UNDER、OVER と続く。UNDER はその VCPU のクレジットの値が正の時でまだクレジットが残っていることを意味する。反対に、OVER はその VCPU のクレジットの値が負の時でクレジットを使い切っていることを意味する。BOOST は I/O のためにブロックされた VCPU を優先するために使われる。クレジットスケジューラはこの優先度を基づいてランキューに VCPU を入れる。

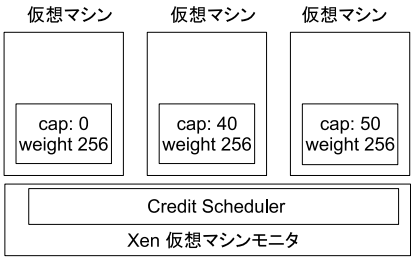


**図 3　Credit Scheduler**

#### 3.2.2 OC-Scheduler

OC-Scheduler は cap、weight というパラメータを持つクレジットスケジューラに debt というパラメータを追加したスケジューラである。仮想マシンがセキュリティ機構をオフロードしている時、そのセキュリティ機構の CPU 使用量がオフロード元の仮想マシンの debt に保存される。たとえば、図 4 のようにセキュリティ機構をオフロードしたとする。オフロードしたセキュリティ機構の CPU 使用率が X %だったら、オフロード元の仮想マシンの debt の値は X となる。

オフロード元の仮想マシンに cap が設定されている場合、cap の値から debt の値を引いた値を cap の値としてクレジットの計算を行う。オフロードしたセキュリティ機構の CPU 使用率分だけオフロード元の仮想マシンの最大 CPU 使用率を下げれば、そのセキュリティ機構とオフロード元の仮想マシンの CPU 使用率の合計は cap に設定された値を越えることはない。配布するクレジットは以下のようにして計算される。

$$\text{配布 } credit = \text{総 } credit \times \frac{cap - debt}{100}$$

一方、weight のみが設定されている場合、オフロードしたセキュリティ機構の CPU 利用率の分だけ weight の値を小さくする。weight に関しては、各仮想マシンの weight の総和に対して、設定された

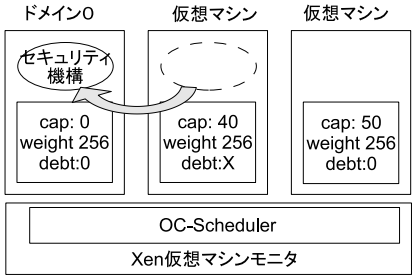


**図 4　OC-Scheduler**

weight の値がどのくらいの割合を占めているかによって配布されるクレジットが決まるためである。

$$\text{配布 } credit = \text{総 } credit \times \frac{weight - \text{総 weight の debt \%分}}{\text{総 } weight}$$

このように debt の値を使って cap と weight の値を調整した上で値で別々にクレジットを計算し、結果の小さいほうをその仮想マシンに配布する。この過程で各仮想マシンに設定された cap と weight の値自体は変更されない。また、debt が 0 の間、つまり、オフロードしたセキュリティ機構が動作していない時は、通常のクレジットスケジューラと同じように動作する。

## 4　実験

セキュリティ機構をオフロードした時の仮想マシン間の性能分離を調べる実験を行った。セキュリティ機構として Snort と Tripwire を用いた。実験は以下の 3 つの場合について行った。(1) セキュリティ機構をオフロードせずにクレジットスケジューラを使った場合、(2) オフロードしてクレジットスケジューラを使った場合、(3) オフロードして OC-Scheduler を使った場合である。

CPU は AMD Athlon 2.2GHz、メモリは 2 GB 、NIC はギガビットイーサ、HDD は SATA 250 GB を搭載したマシンで実験を行った。Xen のバージョンは 3.3.0、ドメイン 0 の OS は Linux 2.6.18-8、ドメイン U の OS は linux 2.6.16.33 である。

### 4.1　Snort

Snort をドメイン 0 で動作させ、オフロード元の仮想マシンを Web サーバとして動作させた。オフロード元の仮想マシンには cap を 40 に設定することで、最大 CPU 使用率 40 %に制限した。ドメイン 0 は cap を 0 に設定し、CPU 利用率に関する制限は行わなかった。weight はともに 256 とした。外部の別のマシンから httperf[6] を利用してウェブサーバを攻撃し、負荷をかける。httperf のリクエストレートは毎秒 4000 リクエストとした。この外部のマシンは、CPU が AMD Athlon 2.2GHz、メモリは 1 GB、NIC はギガビットイーサであった。

Snort にドメイン 0 の仮想インターフェイス vif を監視させることで、ドメイン 0 へのオフロードを可能にした。Xen ではドメイン 0 が仮想マシンのネットワークの送受信を仲介している。これは、ドメイン 0 のみ物理 NIC にアクセスできるためである。ドメイン 0 は各仮想マシンがネットワーク通信できるようにそれぞれに対して vif を提供している。例えば、仮想マシン内のネットワークインターフェイスである eth0 に対応する vif1.0 がドメイン 0 内に作成される。

#### 4.1.1　CPU 使用率

図 5 はオフロード元の仮想マシンと Snort の CPU 使用率の合計の変化である。結果から分かるように、単純にオフロードした場合はオフロード元の仮想マシンの制限である 40 %を越えている。この理由は、オフロード元の仮想マシン内のウェブサーバがその仮想マシンに割り当てられた CPU 資源を使い、オフロードした Snort はオフロード先の仮想マシンの CPU 資源を別に使用したためである。OffloadCage を用いた場合、ほぼ 40 %の制限を守れている。OC-Scheduler ではオフロードしたセキュリティ機構とオフロード元の仮想マシンの CPU 使用量の合計に基づいてスケジューリングを行うため、オフロード元の仮想マシンはオフロードした Snort の分と合わせて最大 40 %となる。一方、オフロードしない場合は、Snort が仮想マシン内にあるため、最大 40 %の制限を守れている。しかし、セキュリティ機構をオフロードした場合よりセキュリティは低下している。

図 6 は Snort 単体の CPU 使用率の変化を示してい

る。単純にオフロードした場合、ドメイン 0 は CPU 使用率の制限がないためオフロードした Snort は CPU 資源を必要な分だけ使用してしまっている。しかし、OffloadCage を用いた場合は、Snort の CPU 使用率は減っている。これは、OC-Scheduler のスケジューリングによりオフロード元の仮想マシンに配布されるクレジットが少なくなるためである。ウェブサーバが単位時間あたりに処理できる httperf のリクエスト数が減り、その結果として Snort が処理しなければならないパケット数も減っている。オフロードしない場合は、同じ仮想マシン内でウェブサーバと Snort が 40 %の資源を分け合うため CPU 使用率が減っている。

図 7 はオフロード元の仮想マシンの CPU 使用率の変化を表している。OffloadCage を用いた場合は、オフロードしたセキュリティ機構の CPU 使用率分だけオフロード元の仮想マシンの CPU 使用率が減少している。単純にオフロードした場合は、与えられた CPU 資源をウェブサーバが十分に使用できるので 40 %に近い値で推移している。オフロードしない場合は、Snort も仮想マシン内に動作しているためウェブサーバと Snort が CPU 使用率の制限である 40 %を使用している。

#### 4.1.2 クレジットの変化

図 8 はオフロード元の仮想マシンに 30ms ごとに配布されるクレジットの変化を示している。Offload-Cage を用いた場合は、オフロード元の仮想マシンに配布されるクレジットが減少している。この減少の度合いはオフロードしたセキュリティ機構の CPU 使用率に対応している。一方、OffloadCage を使わない場合は、同じ値のクレジットが配布されている。これは、クレジットスケジューラにより cap 40、weight 256 の値で計算された値のクレジットが配布されるためである。

#### 4.1.3 スループット

図 9 はオフロード元で動作させた Web サーバのスループットである。OffloadCage を用いた場合は、オフロードしない場合よりスループットが減少している。これは、図 5 に示されているように、オフロードしたセキュリティ機構がオフロードしない場合より

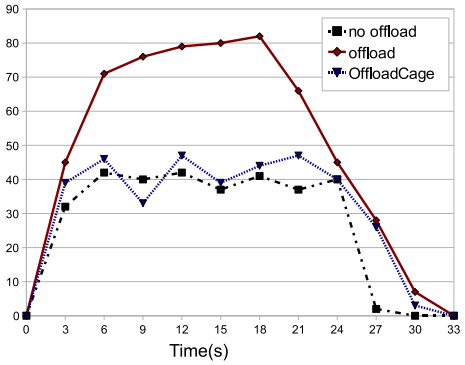


**図 5　Snort のオフロード時の性能分離**

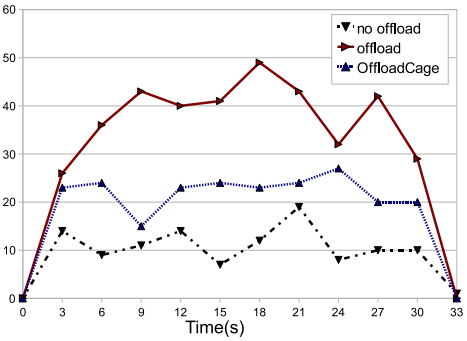


**図 6　Snort の CPU 使用率**

CPU 資源を多く使用しているためである。オフロードしたセキュリティ機構が使う分だけオフロード元の仮想マシンが使える CPU 資源が減るため、その仮想マシンの中で動作するウェブサーバの使用できる CPU 資源もオフロードしない場合より少なくなる。単純にオフロードした場合は逆に、オフロードしたセキュリティ機構の分だけウェブサーバが使用できる CPU 資源が増える。その結果、オフロードしない場合より、ウェブサーバのスループットが増大した。

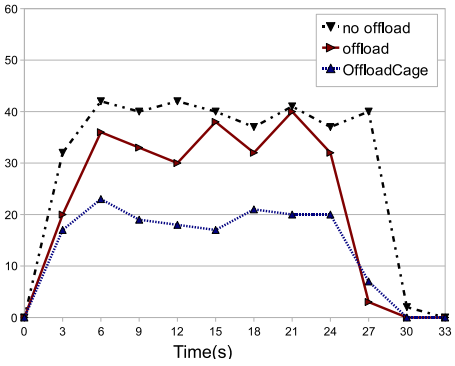


**図 7　オフロード元の仮想マシンの CPU 使用率**

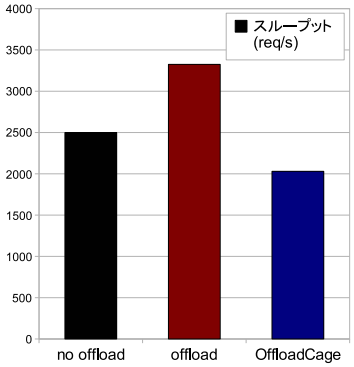


**図 9　ウェブサーバのスループット**

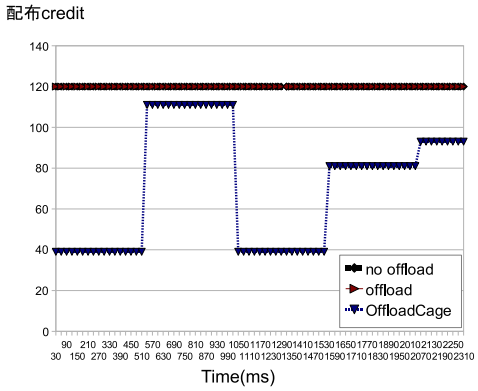


**図 8　オフロード元の仮想マシンに配布される credit**

### 4.2　Tripwire

Tripwire にオフロード元の仮想マシンが使う仮想ディスク内のファイルシステムを検査させることで、ドメイン 0 へのオフロードを実現した。Xen では、イメージファイルを仮想ディスクとして仮想マシンを作成することができる。ループバックデバイスを利用して、このイメージファイルを読み込み専用でマウントすることにより、ドメイン 0 上でオフロード元の仮想マシンのファイルを参照することが可能になる。ただし、ドメイン 0 の OS が仮想ディスク内のファイルシステムを認識できなければならない。ドメイン 0 の OS である Linux は多くのファイルシステムをサポートしているため、この点はあまり問題にならないと考えられる。今回の実験では仮想マシンでも Linux 標準の ext3 ファイルシステムを用いた。

オフロード元の仮想マシンは cap を 50 に設定し、最大 CPU 使用率を 50 %とした。ドメイン 0 は cap を 0 に設定し、CPU 資源に関する制限を行わなかった。weight はともに 256 とした。オフロード元の仮想マシン内では、無限ループするプログラムを実行することで負荷をかけ、最大 CPU 使用率に達している状態にした。Tripwire はそれだけでオフロード元の仮想マシンに設定された最大 CPU 使用率を超えて CPU を使ってしまう場合があるため、cpulimit [9] を使うことで Tripwire は最大 30 %までしか CPU を利用できないように制限した。

図 10 はオフロード元の仮想マシンとオフロードした Tripwire の CPU 使用率の合計である。グラフから分かるように、単純にオフロードした場合、オフロード元の仮想マシンの制限である 50 %を越えている。これは、Tripwire がオフロード元ではなくオフロード先のドメイン 0 の資源を消費したためである。OffloadCage を用いた場合はほぼ 50 %の制限を守れていることが分かる。オフロードしない場合は、Tripwire が仮想マシン中で動作するので 50 %の CPU 使用率を守ることができている。

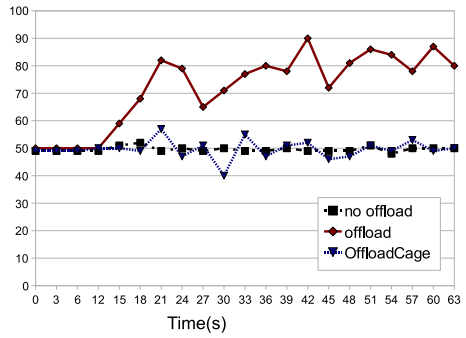


**図 10　Tripwire のオフロード時の性能分離**

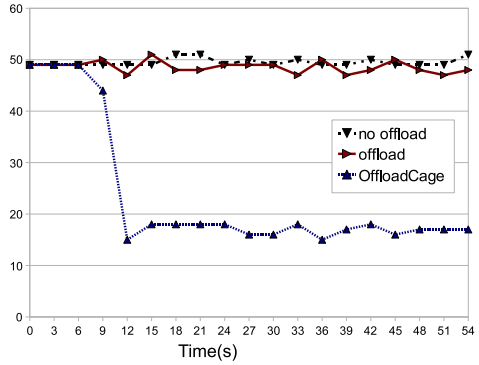


**図 11　オフロード元の仮想マシンの CPU 使用率**

図 11 はオフロード元の仮想マシンの CPU 使用率を示している。OffloadCage を用いた場合は、オフロードした Tripwire の CPU 使用量を考慮してオフロード元の仮想マシンの CPU 使用率が低くなっているのが確認できる。単純にオフロードした場合は、オフロード元の仮想マシン内で行っている無限ループが最大 CPU 使用率まで CPU を使用している。

## 5　関連研究

仮想マシンを利用したセキュリティ機構のオフロードの例として、Livewire [5] や SAccessor [10] などがある。Livewire は仮想マシン内の OS の内部状態を参照することで、侵入検知システムのオフロードを実現している。侵入検知システムが攻撃対象の仮想マシンの外で動作するので、攻撃者から攻撃されにくい。仮想マシンモニタから OS の内部状態を検査するオーバヘッドがあるため、オフロードしない場合に比べると性能が低下する。SAccessor はファイルのアクセス制御をオフロードしている。作業 OS と認証 OS を別々の仮想マシンで動作させ、認証 OS でファイルサーバを動作させることにより、作業 OS 内のファイルのアクセス制御を認証 OS で行う。これにより、作業 OS がクラックされても攻撃者からファイルを守ることができる。

Xen における I/O 処理の性能分離を行うスケジューラとして SEDF-DC [7] がある。Xen のドメイン U の I/O 処理はドメイン 0 を介して行われる。しかし、ドメイン 0 上で行われる処理による CPU 資源の消費は、I/O 処理を行っているドメイン U にカウントされない。そこで SEDF-DC では、ドメイン U のための I/O 処理で使用された CPU 資源をそのドメイン U が使ったものとしてスケジューリングを行う。

LRP [8] はアプリケーション間の性能分離を行っている。ネットワーク処理をよく行うアプリケーションはその処理の大部分をカーネルで行うことになる。しかし、カーネル内のネットワーク処理による資源の消費はそのアプリケーションのものとして適切にカウントされない。LRP では、カーネル内のアプリケーションののネットワーク処理による資源の消費を各アプリケーションに適切にカウントする。

## 6　まとめと今後の課題

本論文ではセキュリティ機構のオフロード時の仮想マシン間の性能分離を実現するシステム OffloadCage を提案した。仮想マシンを利用してセキュリティ機構をオフロードすると、セキュリティは向上するが仮想マシン間の性能分離が実現できなくなる。OffloadCage では、OC-Monitor によりオフロードしたセキュリティ機構による資源の消費を計測して、OC-Scheduler によりオフロードした機構とオフロード元の仮想マシンの資源の使用量を考慮したスケジューリングを行う。

今後の課題としては、オフロードしたセキュリティ機構とオフロード元の仮想マシンとの間での資源の分配について考える必要がある。現在のところ、セキュリティ機構の CPU 使用率はオフロードする前と後で変わっている。これは仮想マシンでのサービスの品質にも影響を与えている。また、監視する資源を CPU 使用量だけではなく、メモリやディスクなどの他の資源にも拡張する。

## 参考文献


[1] Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., and Warfield, A. 2003. Xen and the art of virtualization. SIGOPS Oper. Syst. Rev. 37, 5 (Dec. 2003), 164-177.

[2] Credit Scheduler. http://wiki.xensource.com/xenwiki/CreditScheduler

[3] M. Roesch. Snort - lightweight intrusion detection for networks. In Proceedings of the 13th USENIX System Administration Conference, pages 229-238, 1999.

[4] Kim, G. H. and Spafford, E. H. 1994. The design and implementation of tripwire: a file system integrity checker. In Proceedings of the 2nd ACM Conference on Computer and Communications Security (Fairfax, Virginia, United States). CCS '94. ACM, New York, NY, 18-29. DOI= http://doi.acm.org/10.1145/191177.191183

[5] T Garfinkel, M Rosenblum - A virtual machine introspection based architecture for intrusion detection. In Proceedings of the 10th Annual Symposium on Network and Distributed System Security(NDSS), pages 191-206, Feb 2003.

[6] Mosberger, D. and Jin, T.:httperf - a tool for measuring web server performance, SIGMETRICS Perform. Eval. Rev., Vol.26, No.3, pp.31-37

[7] Gupta, D., Cherkasova, L., Gardner, R., and Vahdat, A. 2006. Enforcing performance isolation across virtual machines in Xen. In Proceedings of the ACM/IFIP/USENIX 2006 international Conference on Middleware (Melbourne, Australia, November 01 - 01, 2006). M. Henning and M. van Steen, Eds. Middleware Conference. Springer-Verlag New York, New York, NY, 342-362.

[8] Druschel, P. and Banga, G. 1996. Lazy receiver processing (LRP): a network subsystem architecture for server systems. SIGOPS Oper. Syst. Rev. 30, SI (Oct. 1996), 261-275. DOI= http://doi.acm.org/10.1145/248155.238786

[9] cpulimit, CPU Usage Limiter for Linux. http://cpulimit.sourceforge.net.

[10] SAccessor：デスクトップ PC のための安全なファイルアクセス制御 滝澤裕二, 光来健一, 千葉滋, 柳澤佳里 情報処理学会論文誌:コンピューティングシステム（ACS）, Vol.1, No.2, pp.275-285, 2008 月 8 月